LABORATORIO VIRTUAL para Enseñanza de Programación

Autores: Fernández, Marcela; Ponzo, Mauricio, Navarría, Salvador
Departamento de Electrónica - Facultad Regional Mendoza
Universidad Tecnológica Nacional
E-mail: mfernand@raiz.uncu.edu.ar
Tel. (061) - 373613 / 298353

Resumen:

Este trabajo presenta un método para lograr una mayor asimilación del alumno sobre el funcionamiento de las estructuras de datos estáticas y dinámicas. Para ello se ha implementado en entorno Windows® una colección de programas que permiten operar, en forma interactiva, sobre pilas, colas, listas y árboles. Se permite al alumno ver la evolución de los datos en la estructura, a medida que va realizando operaciones sobre la misma (inserción, borrado, modificación, recorrido, etc.). Estos conceptos se complementan con una librería de funciones, implementadas en Lenguaje C, que permiten al alumno poder seguir el desarrollo de los datos de las estructuras, a medida que su programa se va ejecutando. Hay funciones específicas para visualizar matrices (de NxM elementos), listas, pilas, colas y secciones de memoria determinadas (como las ocupadas por las variables del programa).

Palabras clave:

Estructuras de datos - Simulación

Introducción

En las cátedras de Informática I e Informática II de la Carrera de Ingeniería Electrónica de la Facultad Regional Mendoza, Universidad Tecnológica Nacional, se detectó la necesidad de transmitir en forma más concisa el significado e interpretación de las estructuras de datos. Esta dificultad en los alumnos se debe a que no es una materia específica de la carrera. El lenguaje de programación que se enseña a los alumnos es el lenguaje C.

Para poder solucionar este inconveniente la cátedra decidió implementar un método más gráfico para poder transmitir a los alumnos estos conceptos, y al mismo tiempo generar una serie de herramientas ("instrumentos") que permitan a los alumnos poder inspeccionar la evolución de sus programas.

Objetivos

Lograr una mayor asimilación del alumno sobre el funcionamiento de las estructuras de datos estáticas y dinámicas.

Asistir al alumno durante el proceso de codificación y depuración de errores.

Marco teórico: estructuras de datos

Se denomina **Estructura lógica** a la forma en que los datos se organizan para constituir una unidad operativa. (Ej. arreglos, listas...)

Tipos de estructuras lógicas

a) Estructuras simples - Tipos de datos:

Datos numéricosDatos alfanuméricos Datos binarios atos puntero *b) Estructuras simples derivadas:*

Variables: Es un objeto, cuyo valor almacenado en una zona de la memoria puede ser

cambiado durante la ejecución del programa.

Constantes: Es un objeto, cuyo valor almacenado en una zona de la memoria permanece inalterado durante toda la ejecución del programa.

Ambas estructuras tienen asociado un "Tipo de Dato".

c) Estructuras complejas:

Son organizaciones de datos que se forman a partir de las estructuras antes mencionadas.

Arreglos
Listas
Pilas
Colas
Arboles
Grafos
Registros
Archivos
Bases de Datos

Características de las estructuras

a) Según la ubicación en la computadora:

Internas: Residen en la memoria principal de la computadora.

Externas: Se encuentran localizadas en un soporte de datos externo (disco rígido, diskette, CD-ROM, etc).

b) Según la forma en que se relacionan los elementos:

Lineales: Cualquiera de sus elementos solamente puede estar enlazado con un sólo elemento anterior y con un único elemento posterior.

No lineales: Cada uno de sus elementos puede tener más de un antecesor o más de un sucesor.

c) Según la posibilidad de modificar el tamaño de memoria:

Estáticas: El tamaño de memoria que requerirá se define antes de la ejecución del programa.

Dinámicas: El tamaño de memoria requerido puede variar durante la ejecución del programa. La memoria se asigna conforme se va necesitando durante la ejecución.

ARREGLOS

Generalidades

Un arreglo es un conjunto finito y fijo de elementos del mismo tipo que se encuentran almacenados en posiciones contiguas de memoria. Este tipo de estructura es estática.

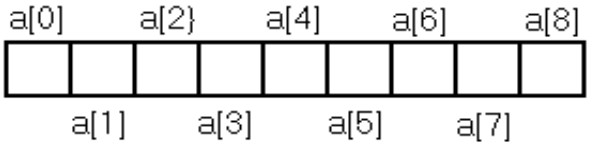
Cada posición puede ser referenciada por un nombre común a todas ellas y por uno o varios índices particulares que indican su posición con respecto al conjunto de elementos. Los índices deben ser números enteros y positivos.

La cantidad de índices necesarios para referirse a un elemento particular de un arreglo determinan la dimensión de dicho arreglo: si tiene un índice se denomina "unidimensional", si tiene dos índices se denomina "bidimensional", si tiene 3, "tridimensional" y si tiene n índices, "n-dimensional".

El tamaño de un arreglo está determinado por el número total de elementos que lo componen y se calcula multiplicando sus índices máximos. Por ejemplo si tenemos un arreglo de **n** filas y **m** columnas, su tamaño será igual a **m** * **n**.

1. Arreglos "unidimensionales" o "vectores"

Un arreglo "unidimensonal" es un conjunto finito y fijo de elementos del mismo tipo que se encuentran almacenados en posiciones contiguas de memoria. Cada elemento se referencia a través del nombre del vector y su índice correspondiente.



Para poder utilizar un arreglo es necesario dimensionarlo, es decir, es necesario reservar memoria para cada uno de sus elementos.

Si se quiere acceder directamente a una dirección de memoria que contiene el valor del elemento k sin pasar por cada una de las posiciones anteriores se utiliza la siguiente fórmula:

$$DirElem[i] = DirElem[0] + TamañoElem \times i$$

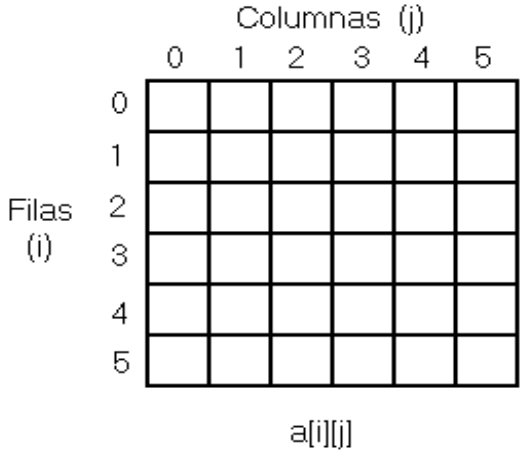
en donde: DirElem[i]: Dirección de memoria a determinar del elemento i.
DirElem[0]: Dirección de memoria del elemento inicial.
TamañoElem: Tamaño de los elementos del vector.
i: Índice del elemento que se está analizando.

Operaciones:

Recorrido: Consiste en acceder en forma sucesiva y consecutiva a los contenidos de cada uno de los elementos del vector.
Inserción: Consiste en introducir en el vector el valor de un elemento.
Borrado: Consiste en eliminar del arreglo el valor de un elemento.
Búsqueda: Consiste en recorrer el vector comenzando por su elemento inicial hasta encontrar el elemento buscado para realizar con él la operación deseada.
Ordenación: Consiste en reorganizar el contenido de un vector siguiendo una secuencia determinada.

2. Arreglos "bidimensionales" o "matrices"

Un arreglo "bidimensonal" es un conjunto finito y fijo de elementos del mismo tipo que se encuentran almacenados en posiciones contiguas de memoria. Cada elemento se referencia a través del nombre del vector y dos índices.



Al dimensionar un arreglo "bidimensional" se está reservando un bloque de memoria. Este bloque puede estar organizado de alguna de las siguientes formas:

- Almacenamiento consecutivo y secuencial de una fila detrás de la otra.
- Almacenamiento consecutivo y secuencial de una columna detrás de la otra.

Si se quiere acceder directamente a una dirección de memoria que contiene el valor del elemento (i,j) sin pasar por cada una de las posiciones anteriores se utilizan las siguientes fórmulas según sea el caso:

Almacenamiento por filas:

$$DirElem[i][j] = DirElem[0][0] + TamañoElem \times (NumCol \times i + j)$$

Almacenamiento por columnas:

$$DirElem[i][j] = DirElem[0][0] + TamañoElem \times (NumFil \times j + i)$$

en donde: DirElem[i][j]: Dirección de memoria del elemento i, j.
DirElem[0][0]: Dirección de memoria del elemento inicial.
TamañoElem: Tamaño de los elementos de la matriz.
NumCol: Número de columnas de la matriz.
NumFil: Número de filas de la matriz.
i, j: Índices del elemento que se está analizando.

Operaciones:

Las operaciones de *Recorrido, Inserción, Borrado, Búsqueda y Ordenamiento* son iguales a las descriptas para "Vectores" sólo que se debe tener en cuenta que los índices involucrados son dos.

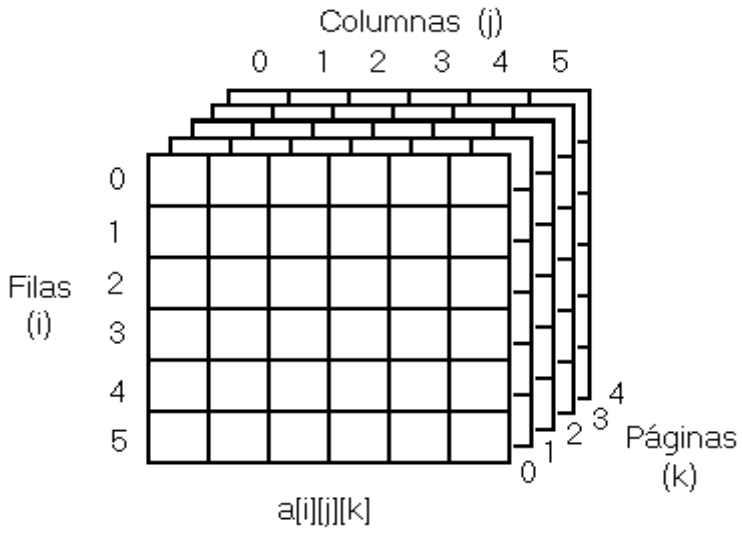
3. Arreglos "tridimensionales" o "poliedros"

Un arreglo "tridimensonal" es un conjunto finito y fijo de elementos del mismo tipo que se encuentran almacenados en posiciones contiguas de memoria. Cada elemento se referencia a través del nombre del vector y tres índices.

Al dimensionar un arreglo "tridimensional" se está reservando un bloque de memoria. Este bloque puede estar organizado de alguna de las siguientes formas:

- Almacenamiento consecutivo y secuencial de una fila detrás de la otra y de una página detrás de la otra.
- Almacenamiento consecutivo y secuencial de una columna detrás de la otra y de una página detrás de la otra.

Si se quiere acceder directamente a una dirección de memoria que contiene el valor del elemento (i,j,k) sin pasar por cada una de las posiciones anteriores se utilizan las siguientes fórmulas según sea el caso:



Almacenamiento por filas:

$$DirElem[i][j][k] = DirElem[0][0][0] + TamañoElem \times (NumFil \times NumCol \times k + NumCol \times i + j)$$

Almacenamiento por columnas:

$$DirElem[i][j][k] = DirElem[0][0][0] + TamañoElem \times (NumFil \times NumCol \times k + NumFil \times j + i)$$

en donde:
DirElem[i][j][k]: Dirección de memoria del elemento i, j, k.
DirElem[0][0][0]: Dirección de memoria del elemento inicial.
TamañoElem: Tamaño de los elementos del arreglo.
NumCol: Número de columnas del arreglo.
NumFil: Número de filas del arreglo.

i, j,k: Índices del elemento que se está analizando.

Operaciones:

Las operaciones de *Recorrido, Inserción, Borrado, Búsqueda y Ordenamiento* son iguales a las descriptas para "Vectores" sólo que se debe tener en cuenta que los índices involucrados son tres.

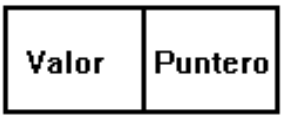
LISTAS

Una lista lineal es un conjunto de elementos de un tipo determinado, relacionados entre sí de modo tal que cada elemento sólo puede tener un único elemento anterior y un único elemento posterior.

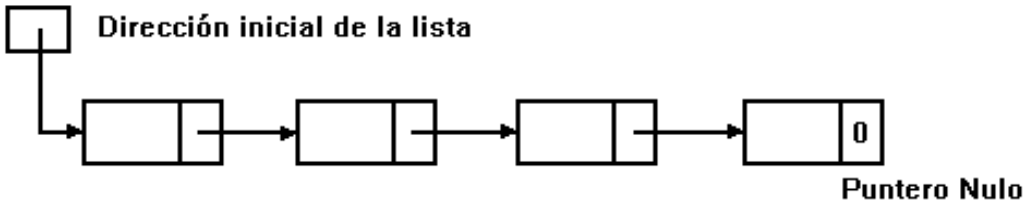
Dependiendo de su implementación las listas pueden ser estáticas o dinámicas, internas o externas.

1. Listas encadenadas:

Es una lista lineal en la que el orden de sus elementos se establece mediante punteros. Esto significa que cada elemento contiene un valor y la dirección del elemento siguiente. Los elementos de las listas reciben el nombre de "Nodos".



Las listas encadenadas comienzan con un elemento inicial cuyo "puntero de cabecera" debe ser leído por el programa de alguna parte y termina con un elemento final cuyo segundo campo contiene un puntero nulo para indicar que este elemento es el último de la lista.



Los distintos elementos de la lista pueden estar almacenados en posiciones no adyacentes de memoria. Por este motivo estas listas pueden crecer indefinidamente mientras haya memoria libre. La longitud de una lista encadenada depende del número de elementos que contenga.

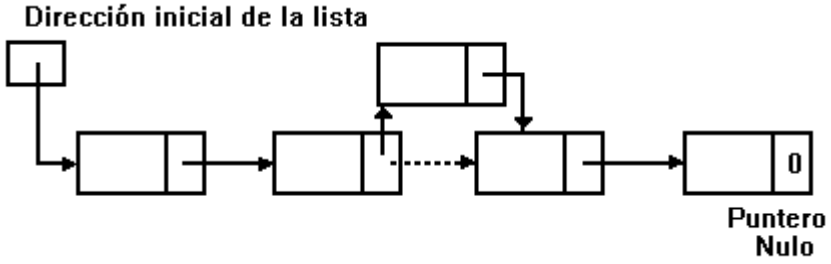
Operaciones:

Recorrido: Consiste en acceder secuencialmente a cada uno de los elementos de la lista.

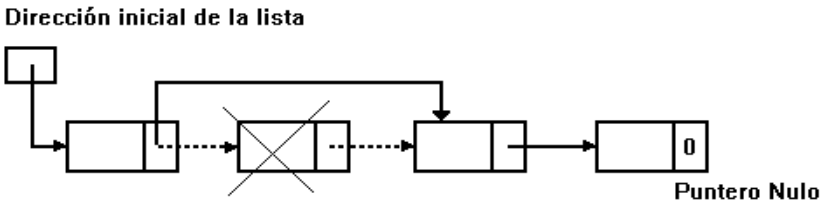
Búsqueda: Consiste en recorrer la lista comparando el valor del elemento leído con el dato a localizar hasta encontrarlo o bien hasta terminar el recorrido.

Inserción: Consiste en recorrer la lista hasta encontrar el elemento detrás del que se desea insertar el nuevo elemento, asignar su puntero al nuevo elemento y cambiar el propio por un puntero que indique la dirección del nuevo elemento.

Borrado: Consiste en recorrer la lista hasta encontrar el elemento que se desea borrar, cambiar el puntero del elemento anterior para que indique la dirección del

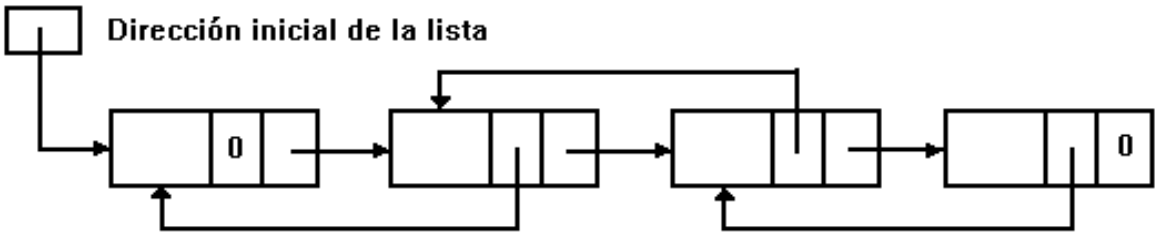


elemento siguiente al que se desea borrar.



2. Listas doblemente encadenadas:

Una lista doblemente encadenada es una lista lineal en la que el orden de sus elementos se establece utilizando 2 punteros: uno de ellos se utiliza para indicar la dirección del elemento anterior y el otro para indicar la dirección del elemento siguiente.

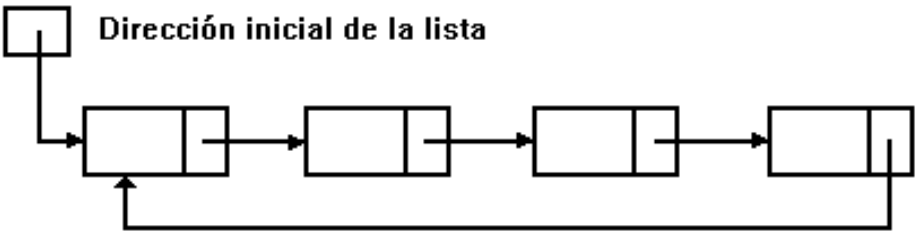


Operaciones:

Las operaciones de *Recorrido, Búsqueda, Inserción y Borrado* se realizan de la misma manera que con las listas simples con la salvedad de que ahora existen dos punteros a considerar.

3. Listas circulares:

Una lista circular es una lista encadenada cuyo último elemento contiene un puntero que indica la dirección del elemento inicial de la lista.

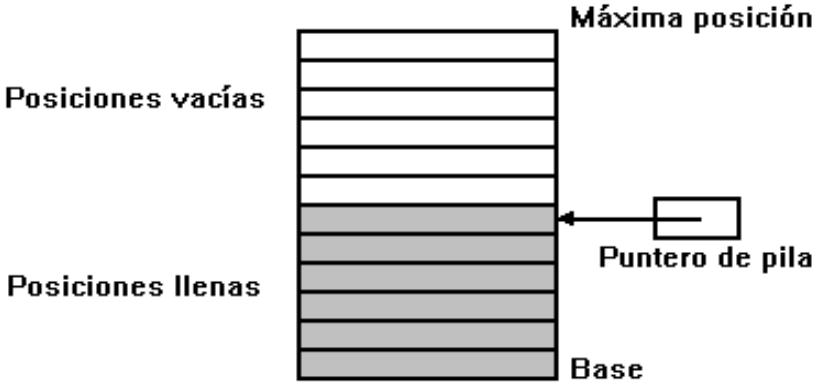


Presenta la ventaja que partiendo desde cualquiera de sus elementos se puede recorrer la lista completa. Pero para evitar que estos recorridos se conviertan en bucles infinitos las listas circulares poseen un "nodo de cabecera" que indica cuál es el primer elemento de la lista.

4. Pilas

Una pila es una lista lineal en la que solamente se puede acceder a sus elementos por un extremo de la misma.

Las pilas se conocen también como estructuras LIFO (Last Input - First Output), esto significa que el último elemento que ha entrado en ella será el primero en salir.

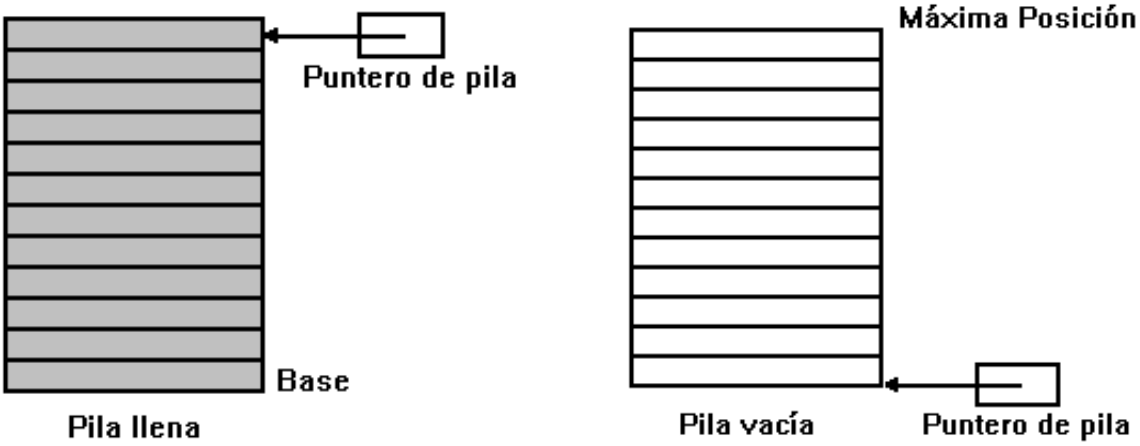


El "puntero de pila" señala o bien la última posición ocupada, o bien la primera posición libre de la misma dependiendo de la implementación.

Los elementos de la pila no se mueven con las entradas o las salidas de los mismos. Permanecen fijas en las posiciones de memoria que se les ha asignado a su entrada. Sólo el "puntero de pila" cambia su valor para señalar siempre la posición del último elemento de la pila.

Por lo general, es necesario reservar una cierta cantidad de posiciones de memoria para ser utilizadas con la pila de manera tal que garantice una mínima cantidad de elementos en la estructura.

Si el "puntero de pila" es igual a la máxima posición de memoria permitida, la pila está llena. En cambio, si el "puntero de pila" es igual a la "base", la pila está vacía.



Cada vez que el "puntero de pila" se incrementa o decrementa lo hace en una "unidad de memoria", es decir, se incrementa o decrementa en el conjunto de posiciones de memoria necesarias para almacenar un elemento de la pila.

Operaciones:

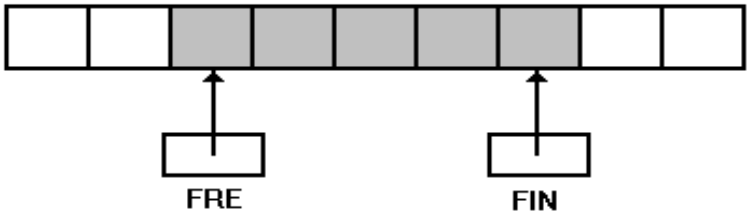
Poner un elemento: Requiere la ejecución de los siguientes pasos:
a) Comprobar si la pila está llena.
b) Si no está llena, incrementar en una unidad el "puntero de pila".
c) Introducir el nuevo elemento en la posición indicada por el "puntero de pila".

Quitar un elemento: Requiere la ejecución de los siguientes pasos:
a) Comprobar si la pila está vacía.
b) Si no está vacía, extraer el elemento señalado por el "puntero de pila".
c) Decrementar el "puntero de pila" en una unidad.

5. Colas

Una "Cola" es una lista lineal en la que las extracciones se realizan sólo por el principio de la lista y las inserciones sólo por el final de la misma.

Las "colas" se conocen también como estructuras FIFO (First Input - First Output), esto significa que el primer elemento que ingrese será el primer elemento en salir.



En este tipo de estructura sólo se pueden realizar 2 operaciones (introducir o extraer elementos) para lo cual es necesario utilizar 2 punteros: INI (que señala el principio de la cola) y FIN (que señala el final de la misma).

Durante las operaciones los elementos de la cola permanecen fijos en las posiciones de memoria que se les ha asignado a su entrada. Sólo los punteros cambian su valor indicando el primero y el último elemento de la cola respectivamente.

La "cola" está vacía si se cumple que: INI = FIN = 0

La "cola" está llena si se cumple alguna de estas condiciones:
INI = FIN + 1, o INI = FIN + 1 - N, siendo N el número de elementos de la cola.

Cada vez que los punteros varían lo hacen en una "unidad de memoria", es decir, en el conjunto de posiciones de memoria necesarias para almacenar un elemento de la cola. Si cualquiera de los dos punteros señala el final de la "cola", un incremento supone indicar la primera posición de la "cola".

Operaciones:

Agregar un elemento: Requiere la ejecución de los siguientes pasos:
a) Comprobar si la "cola" está llena.
b) Si no la está, incrementar FIN en una unidad.
c) Introducir el nuevo elemento en la posición indicada por FIN.

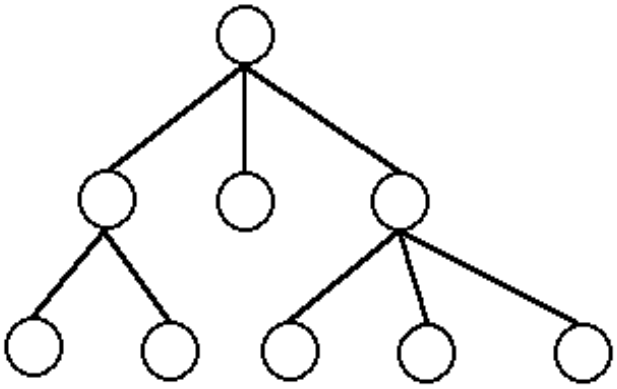
Extraer un elemento: Requiere la ejecución de los siguientes pasos:
a) Comprobar si la "cola" está vacía.
b) Si no la está, extraer el elemento señalado por INI.
c) Incrementar INI en una unidad.

ARBOLES

Generalidades

Un árbol es una lista no lineal formada por un conjunto de elementos del mismo tipo jerarquizados de la siguiente forma:

- Cada uno de sus elementos tiene un solo antecesor y puede tener un sucesor, varios o ninguno.
- Existe un elemento llamado "raíz" que no tiene ningún antecesor.

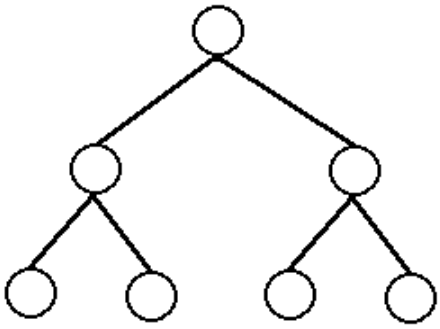


Terminología

Raíz: Elemento sin antecesor del cual se derivan todos los demás.
Nodo: Cada uno de los elementos que componen el árbol.
Antecesor: Elemento del cual se derivan otros. Tambien recibe el nombre de "ascendiente" o "padre".
Sucesor: Elemento derivado de un "antecesor" o "padre". Tambien recibe el nombre de "descendiente" o "hijo".
Hoja: Elemento que no tiene ningún sucesor. También recibe el nombre de "nodo terminal".
Camino: Secuencia de enlaces entre nodos consecutivos.

Rama: Camino que termina en una "hoja".
Nivel: Número de orden asignado a un nodo. El nodo raíz tiene nivel 0. Un nodo sucesor tiene el nivel de su "antecesor" mas uno.
Altura o profundidad de un nodo: Longitud del camino más largo desde ese nodo hasta una hoja.

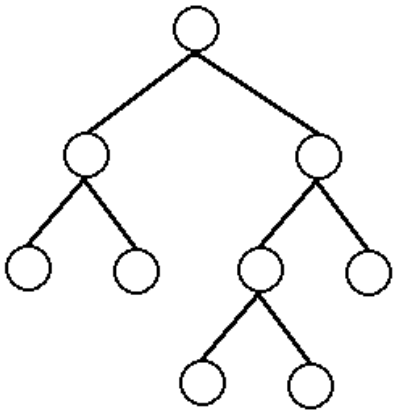
Arbol binario



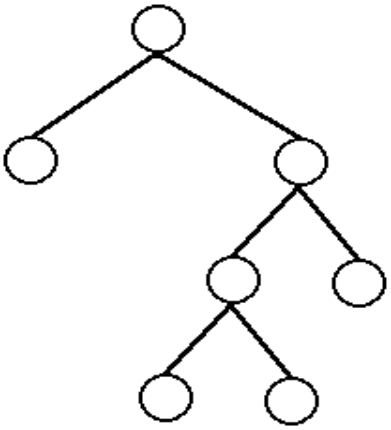
Un árbol binario es una lista no lineal formada por un conjunto de elementos del mismo tipo organizados del siguiente modo:

- Cada uno de sus elementos tiene un solo antecesor y puede tener uno, dos o ningún sucesor.
- Existe un elemento llamado "raíz" que no tiene ningún antecesor.

Un árbol binario está **equilibrado** o **balanceado** si la altura de los dos subárboles del nodo raíz son iguales o difieren en una unidad.



Árbol equilibrado



Árbol no equilibrado

Un árbol binario está **completo** si todos sus nodos tienen dos hojas a excepción de las hojas. Un árbol binario está **ordenado** si el valor de cualquier nodo es mayor que el valor de cualquier nodo del subárbol izquierdo y menor que el valor de cualquier nodo del subárbol derecho.

Operaciones:

Inserción: Requiere la ejecución de los siguientes pasos:
a) Introducir el valor a insertar.
b) Comparar el valor a insertar con el de la "raíz". Si es mayor, ir al subárbol derecho, caso contrario ir al subárbol izquierdo.
c) Proceder del mismo modo hasta que se dé alguno de los siguientes casos:
- El valor a insertar es igual al valor de la "raíz" del subárbol: insertar a la izquierda.
- Se alcanzó una "hoja": si el valor a insertar es menor que el valor del nodo "hoja" introducir el nuevo valor a la izquierda, caso contrario introducirlo a la derecha del nodo "hoja".

Borrado: Requiere la ejecución de los siguientes pasos:
a) Introducir el valor a borrar.
b) Comparar el valor a borrar con el de la "raíz". Si es mayor, ir al subárbol derecho, caso contrario ir al subárbol izquierdo.
c) Proceder del mismo modo hasta encontrar un nodo cuyo valor sea el que se tiene

que borrar.

d) Una vez localizado el nodo a borrar se puede producir alguno de los siguientes casos:
   - El nodo a borrar no tiene hijos: se elimina directamente.
   - El nodo a borrar tiene un hijo: el hijo pasa a ocupar el lugar del padre.
   - El nodo tiene dos hijos: se reemplaza el nodo a eliminar por el nodo de valor más próximo y se borra este último del lugar que ocupa. Si este nodo está en el subárbol derecho del nodo a eliminar se accede al primer nodo y se recorre hasta el final el subárbol izquierdo. Si está en el subárbol izquierdo, se accede a su primer nodo y se recorre hasta el final el subárbol derecho.

Búsqueda: Requiere la ejecución de los siguientes pasos:

a) Introducir el valor a buscar.
b) Comparar el valor a buscar con el de la "raíz". Si es mayor, ir al subárbol derecho, caso contrario ir al subárbol izquierdo.
c) Proceder del mismo modo hasta encontrar el valor buscado o bien hasta llegar a una "hoja".

Recorrido de un árbol binario: Consiste en acceder a cada uno de los nodos del árbol en su totalidad sin visitar dos veces el mismo elemento. Para esto se requiere la ejecución de los siguientes pasos:

a) Acceder al nodo "raíz".
b) Recorrer los nodos del subárbol izquierdo.
c) Recorrer los nodos del subárbol derecho.

Estos tres pasos se pueden realizar en distinto orden, lo cual genera tres tipos de recorrido:

Recorrido pre-orden:

a) Acceder al nodo "raíz".
b) Recorrer los nodos del subárbol izquierdo (en pre-orden).
c) Recorrer los nodos del subárbol derecho (en pre-orden).

Recorrido in-orden:

a) Recorrer los nodos del subárbol izquierdo (en in-orden).
b) Acceder al nodo "raíz".
c) Recorrer los nodos del subárbol derecho (en in-orden).

Recorrido post-orden:

a) Recorrer los nodos del subárbol izquierdo (en post-orden).
b) Recorrer los nodos del subárbol derecho (en post-orden).
c) Acceder al nodo "raíz".

Conversión de un árbol general en binario:

Las operaciones con árboles binarios son mas sencillas de implementar que las operaciones con los árboles generales. Por este motivo es conveniente convertir los árboles generales en binarios para luego poder procesarlos.

Las reglas para su conversión son las siguientes:

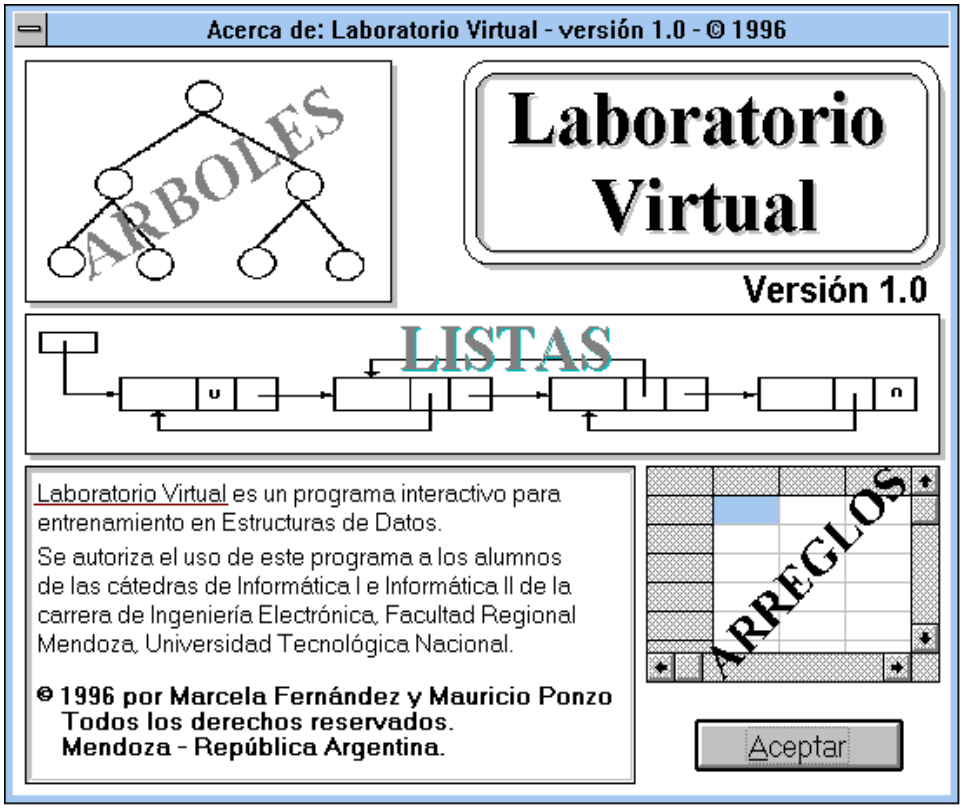
a) El árbol binario que se obtiene debe tener los mismos nodos del árbol general.
b) El árbol binario tiene el mismo nodo raíz que el árbol general.

Para generar el árbol binario se procede del siguiente modo:

- Se enlaza la "raíz" con su nodo hijo situado más a la izquierda.
- Se enlaza este nodo hijo con todos sus hermanos.
- Se repite el mismo proceso para todos los niveles.
- Se gira el diagrama resultante hasta que se distingan perfectamente los elementos izquierdo y derecho de cada nivel.

Entrenamiento Interactivo

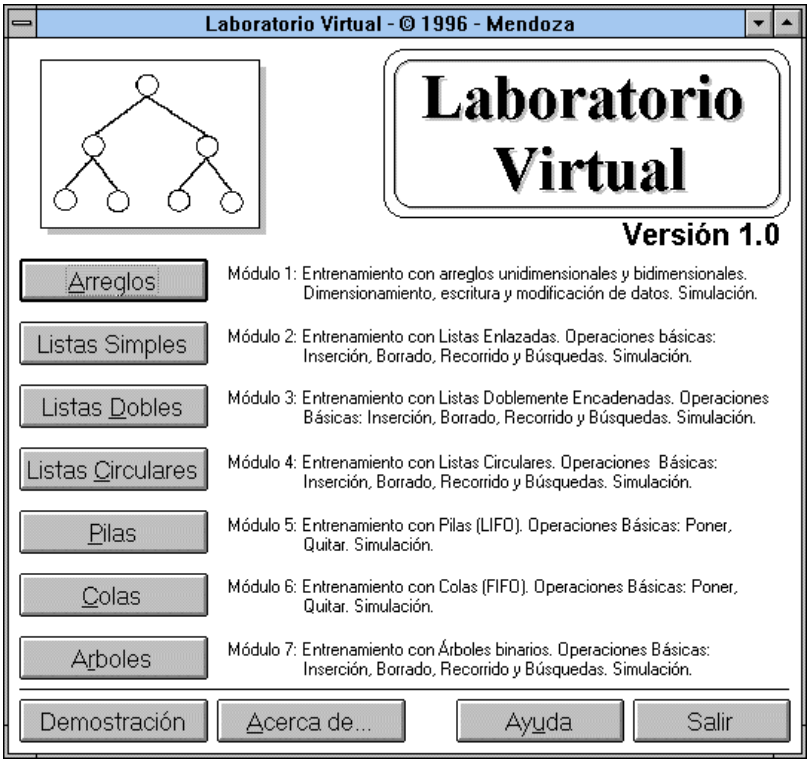
Los programas interactivos están desarrollados para entorno Windows®, y le permiten al alumno crear una estructura de datos, insertarle datos, borrarlos, modificarlos, recorrer la estructura, ordenarla, etc., verificando los resultados de cada operación realizada, de manera de comprender el significado de cada una de las operaciones al poder visualizar permanentemente los valores actuales en la pantalla. Se ha elegido el entorno Windows® gracias a su facilidad de operación y a la posibilidad de incorporar una forma intuitiva de operación que no requiera un entrenamiento adicional.



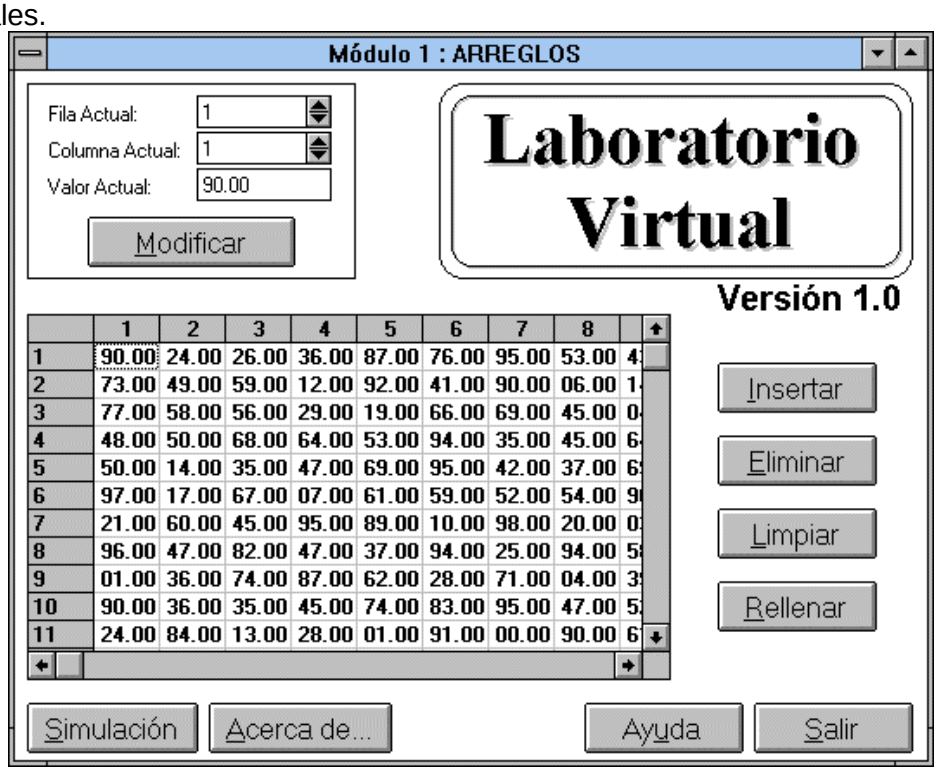
Esta pantalla es el menú principal, y permite ejecutar cada uno de los programas desarrollados, a la vez que centraliza el control de la aplicación. Como es natural en el entorno Windows, toda la operatoria del programa se realiza a través del "ratón", aunque también se ha previsto el control del mismo desde el teclado.

Para acceder a cada "botón" con el teclado es suficiente con presionar la tecla [Alt] y la letra que se encuentra subrayada en cada opción (ambas teclas simultáneamente).

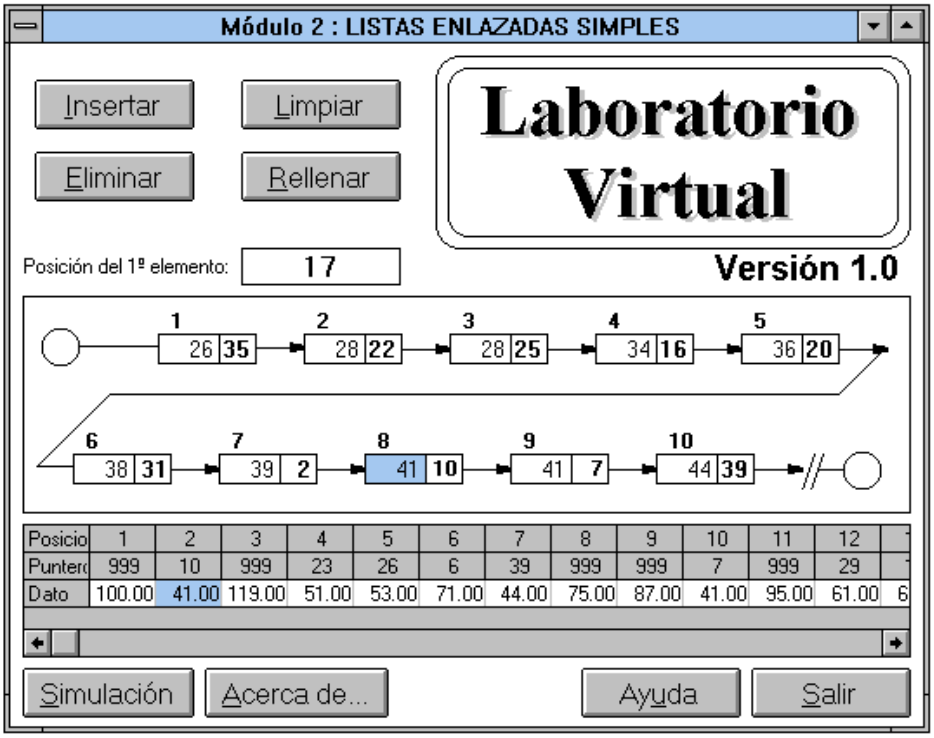
La primera opción permite el entrenamiento con arreglos, observándose una pantalla perteneciente a los arreglos



bidimensionales.



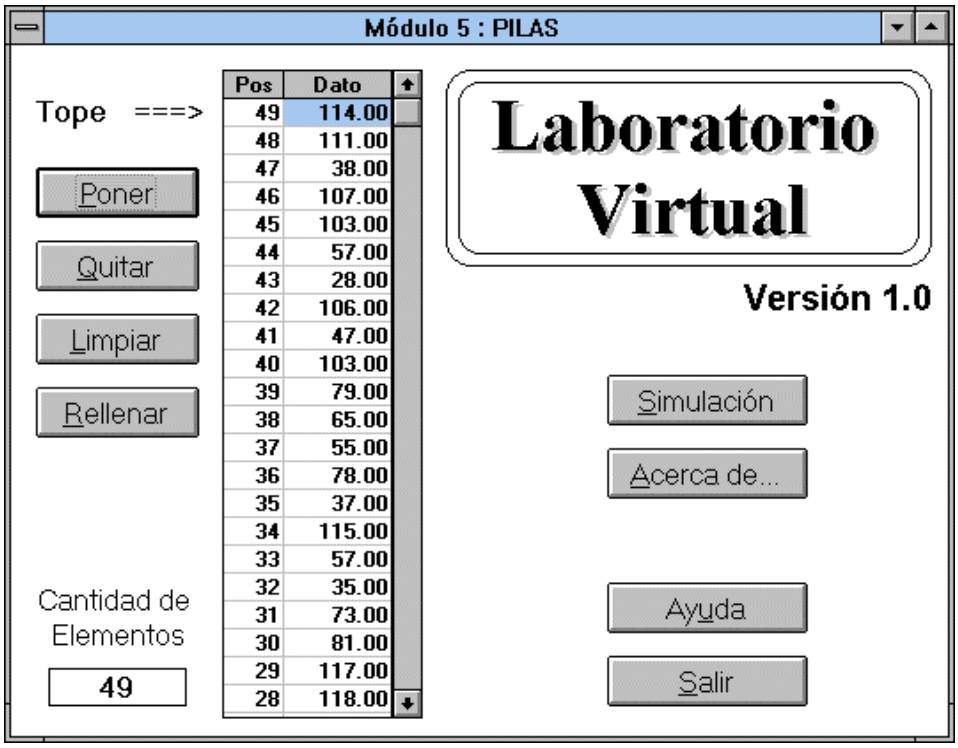
Otra de las opciones, correspondiente a Listas Simplemente Enlazadas, puede observarse a continuación.



Se incluye, como parte del entrenamiento, una animación detallada de los procesos de inserción y borrado de elementos en la lista, para que el alumno comprenda la mecánica de asignación de punteros.

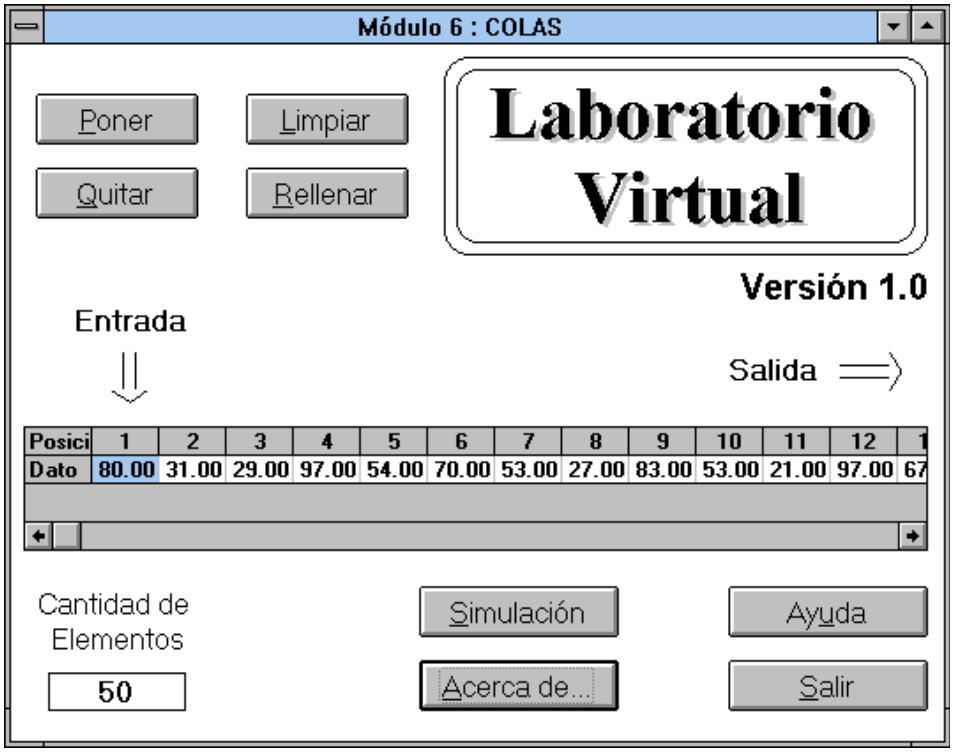
Para las otras opciones de listas (Listas Doblemente Enlazadas y Listas Circulares) el alumno dispone de programas similares, con las mismas características que éste.

En la opción de Pilas, se obtiene la siguiente pantalla.



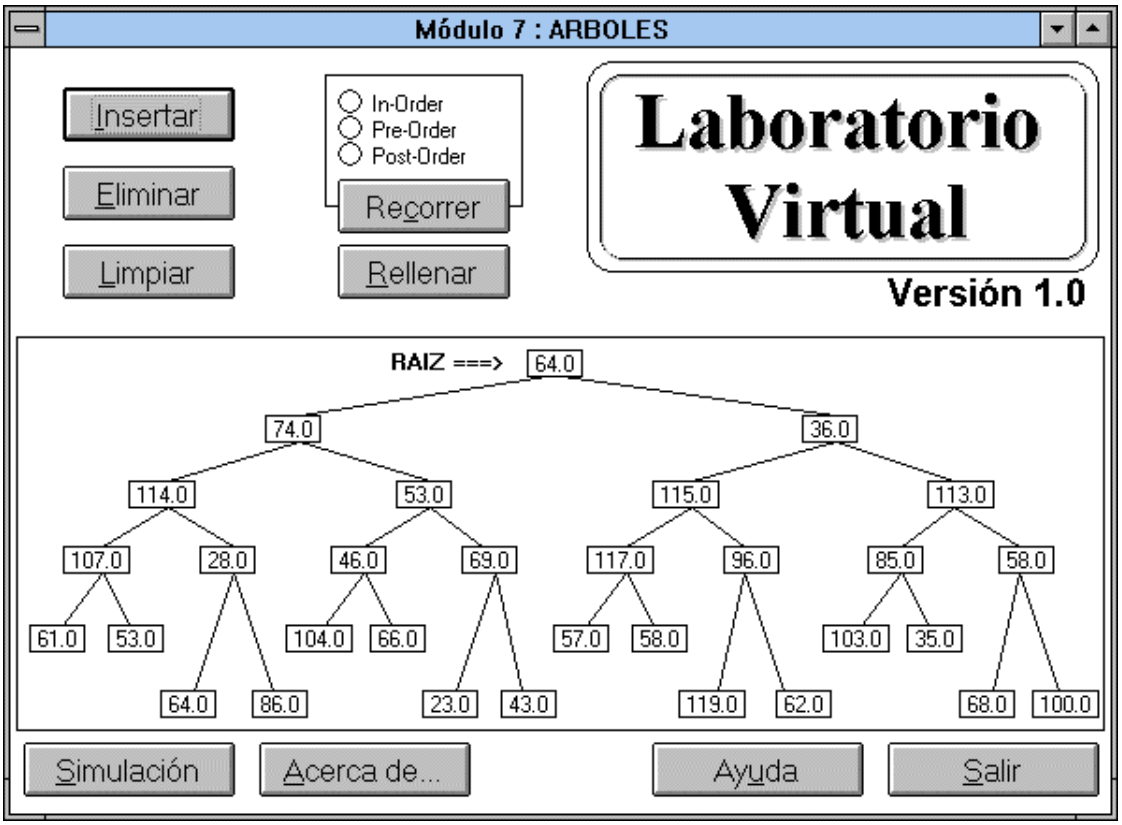
Allí el alumno puede ensayar las operaciones de poner y quitar elementos. El programa mostrará un mensaje especial para indicar que la pila ha alcanzado su límite superior (pila llena), y otro mensaje para el caso en que la pila haya quedado vacía.

En la opción de colas, el alumno verá la siguiente pantalla:



Con este programa pueden ensayarse todas las operaciones correspondientes a la cola. El programa, además, indicará al alumno cuando la cola haya quedado vacía.

En la última opción, correspondiente a Árboles, el alumno encontrará la siguiente pantalla:

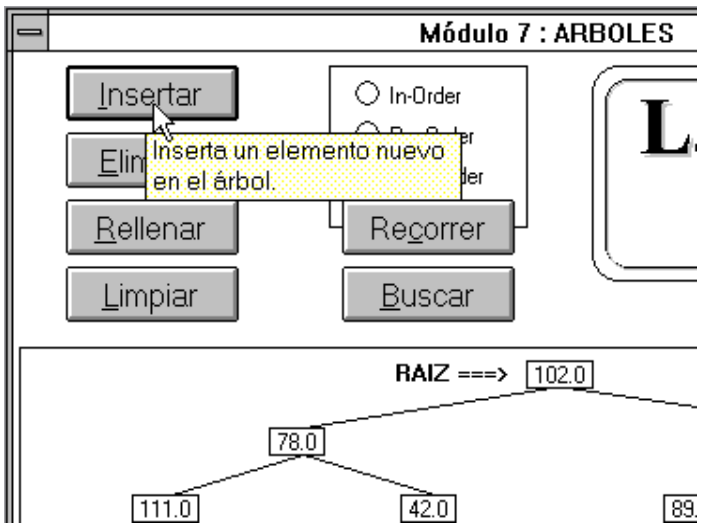


En ella puede visualizarse la estructura completa del árbol (para un árbol binario). El alumno puede interactuar con el programa para alterar el contenido de alguno de sus nodos, así como borrarlo, o insertar otro.

Se incluyen animaciones detalladas paso a paso de los diferentes recorridos, y del proceso de búsqueda, con la finalidad de que el alumno interprete cada una de los pasos necesarios para llevar a cabo cada operación.

Cabe mencionar que cada uno de estos módulos contiene una opción especial, denominada "Simulación", en donde el programa realizará (paso a paso) una serie de operaciones (con datos aleatorios) de manera tal que el alumno pueda comprender los objetivos de la estructura planteada, a la vez que interpreta el significado de cada operación.

Puede realizarse un recorrido rápido por las simulaciones de todos los módulos si se elige la opción "Demostración" del menú principal.



Además de las animaciones, y para que el programa no presente complicaciones adicionales al alumno (cuya única finalidad es aprender sobre las estructuras de datos), todos los módulos contienen "ayudas emergentes" que explican rápidamente en que consiste la operación indicada.

El programa también cuenta con una opción de "Ayuda". Esta opción cuenta con información sobre cómo utilizar el programa, pero lo más importante, es que además contiene los fundamentos teóricos de cada una de las estructuras analizadas. Esto posibilita al alumno un entrenamiento teórico-práctico, a la vez que permite ensayar cada opción a medida que se la

estudia. Esta ayuda se encuentra en formato HLP de Windows, con lo cual se posee el beneficio adicional del hipertexto, que permite mayor dinamismo en la asimilación de conocimientos.

Laboratorio Virtual

Las librerías implementadas están desarrolladas para entorno DOS, en modo texto, y le permiten al alumno poder visualizar los resultados de su programa, a medida que éste se va ejecutando, sin necesidad de añadir una complejidad adicional (que consideramos innecesaria) a los algoritmos por él desarrollados. Estas librerías funcionan como "instrumentos" que le permiten al alumno "medir" su programa mientras éste va funcionando, haciendo de cuenta que ha colocado su programa en un "laboratorio" para poder ensayarlo.

El alumno se encuentra con una típica ventana de TurboVision (para DOS) en la cual puede observar el contenido de un arreglo, una lista, una pila, o una cola. La librería de árboles todavía está en desarrollo.

Para poder activarla sólo debe incluir el archivo de cabecera (.H) correspondiente, invocar la función de inicialización, la cual asigna la estructura a analizar, así como también su tamaño.

Para inspeccionar o visualizar el contenido, es suficiente con llamar la función de visualización (VerLista, VerMatriz, etc) en el momento en que se requiera. La rutina hace aparecer la ventana sobre la pantalla y permite su recorrido (con cursores, o ratón) hasta que se presione la tecla [ENTER], o se haga doble-click con el ratón.

Antes de finalizar el programa se debe liberar la visualización con la función correspondiente, para que se vuelva a compactar la memoria asignada en forma dinámica.

De esta manera se puede observar la evolución de una estructura de datos en la memoria a medida que el programa se ejecuta. Para el alumno representa una herramienta muy valiosa, puesto que observa la estructura en una forma muy similar a la que vio mientras estudiaba la teoría.

Como único inconveniente de estas librerías debemos mencionar que la memoria requerida, y consumida, es elevada. No consideramos que sea un inconveniente crítico, pues la finalidad de estas librerías es el aprendizaje y entrenamiento de los alumnos, y los programas que desarrollarán no serán tan extensos como para impedir su ejecución en una computadora de características medias.

Trabajos futuros

Se está analizando la posibilidad de ampliar estas librerías para incorporar "instrumentos" que permitan simular el comportamiento de dispositivos externos. Debido a que la especialidad es Electrónica, se pretende que los alumnos puedan utilizar las capacidades de Entrada/Salida de la computadora para poder accionar actuadores y visualizadores externos. Como esto incorpora dificultades que son ajenas a la asignatura, se pretende desarrollar "instrumentos", semejantes a las librerías anteriormente mencionadas, que permitan interceptar las lecturas y escrituras en los puertos de entrada/salida y simular los dispositivos a ellos conectados; como por ejemplo, visualizadores de siete segmentos, LEDs, motores, relés, etc. También se desarrollaría un kit de placas didácticas para poder llevar a la práctica lo que se ha ensayado en el "laboratorio virtual"

Conclusiones

La utilización de métodos interactivos mejora la capacidad de asimilación de los alumnos sin la necesidad de que los mismos tengan que verse involucrados en toda la problemática que implica ***aprender*** un lenguaje. De este modo abstraen el concepto de estructuras de datos de la implementación en sí misma.

En una segunda etapa, los alumnos ya dedicados al aprendizaje del lenguaje pueden trabajar en este laboratorio virtual para implementar las estructuras antes aprendidas encontrándose con un entorno amigable e intuitivo que les permite relacionar el entrenamiento previo con la implementación real de dichas estructuras.

El desarrollo e implementación de ***Laboratorios Virtuales*** le brinda al alumno una transición más suave entre la teoría y la práctica asistiéndolo durante el proceso de depuración de errores, como así también reforzando, y corrigiendo si fuera necesario, los conceptos teóricos adquiridos. El tiempo total de aprendizaje se reduce considerablemente al utilizar estas herramientas.

La utilización de ***Laboratorios Virtuales*** durante el proceso enseñanza-aprendizaje brinda la posibilidad que el desarrollo de las clases sea teórico-práctico.

Bibliografía

- "*Estructura de la información*", Laporta, G, Ed. McGraw Hill, 1992.
- "*Algoritmos y estructuras de datos*", Wirth, N, Ed. Prentice Hall, México, 1987.
- "*Estructuras de datos y algoritmos*", Aho, A. y otros, Ed. Addison-Wesley, México, 1988.
- "*Metodología de la programación*", Rodriguez Almeida, M.A., Ed. McGraw Hill, 1993.
- "*Diseño y manejo de estructuras de datos en C*", Villalobos, Ed. McGraw Hill, México, 1995.

Ing. Marcela Fernández
Sr. Mauricio Ponzo
Ing. Salvador Navarría
Mendoza, junio de 1997